This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 :		(11) International Publication Number:	WO 98/21648
G06F 9/00	A1	(43) International Publication Date:	22 May 1998 (22.05.98)

(21) International Application Number: PCT/US97/20627

(22) International Filing Date: 13 November 1997 (13.11.97)

(30) Priority Data:

08/749,926	13 November 1996 (13.11.96)	US
08/927,922	11 September 1997 (11.09.97)	US
08/964,751	5 November 1997 (05.11.97)	US

(71) Applicant *(for all designated States except US)*: PUMA TECHNOLOGY, INC. [US/US]; 2550 North First Street #500, San Jose, CA 95131 (US).

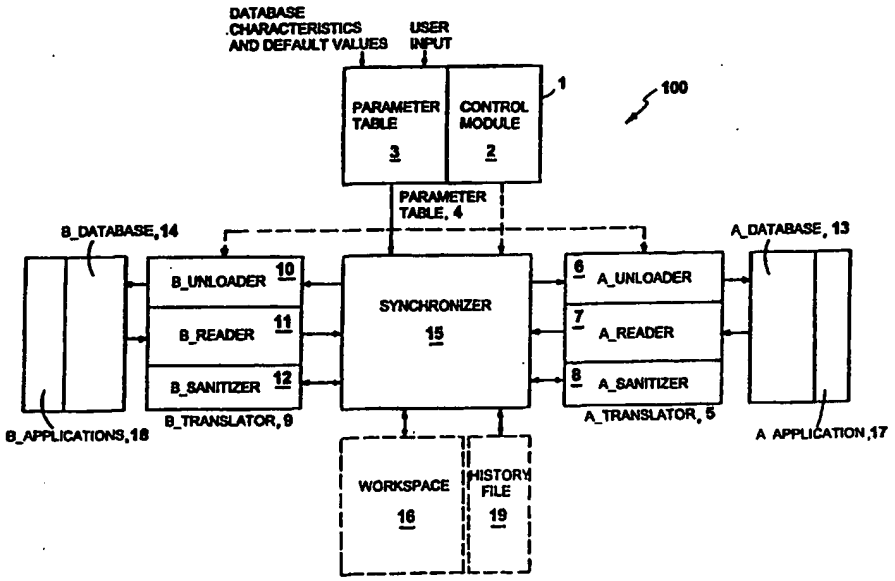
(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: BOOTHBY, David, J. [US/US]; 12 Thoreau Drive, Nashua, NH 03062 (US). DALEY, Robert, C. [US/US]; 13 Middle Dunstable Road, Nashua, NH 03062 (US). MARIEN, John, R. [US/US]; 23 Royal Crest Drive, No. 12, Nashua, NH 03062 (US).

(74) Agent: LEE, G., Roger; Fish & Richardson P.C., 225 Franklin Street, Boston, MA 02110-2804 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZW, ARIPO patent (GH, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(54) Title: SYNCHRONIZATION OF DATABASES



(57) Abstract

A computer implemented method and a computer program for synchronizing a first (13) and a second database (14), where data is provided for keeping track of whether the records of the first database have been added or changed since a previous synchronization. Based on the data reflecting whether the records of the first database have been added or changed since a previous synchronization, it is determined whether the records of the first database have been changed or added since the previous synchronization. The representative record is stored in a history file (19) which contains records reflecting the contents of records of the databases at the time of a previous synchronization.

OCID: <WO___9821648A1_I_>

SYNCHRONIZATION OF DATABASES

Background

This invention relates to synchronizing incompatible databases.

Databases are collections of data entries which are organized, stored, and manipulated in a manner specified by applications known as database managers (hereinafter also referred to as "Applications"; hereinafter, the term "database" also refers to a database manager combined with a database proper). The manner in which database entries are organized in a database is known as the data structure of a database. There are generally two types of database managers. First are general purpose database managers in which the user determines (usually at the outset, but subject to future revisions) what the data structure is. These Applications often have their own programming language and provide great flexibility to the user. Second are special purpose database managers that are specifically designed to create and manage a database having a preset data structure. Examples of these special purpose database managers are various scheduling, diary, and contact manager Applications for desktop and handheld computers. Database managers organize the information in a database into records, with each record made up of fields. Fields and records of a database may have many different characteristics depending on the database manager's purpose and utility.

Databases can be said to be incompatible with one another when the data structure of one is not the same as the data structure of another, even though some of the content of the records is substantially the same. For example, one database may store names and addresses in the following fields: FIRST_NAME, LAST_NAME, and

)OCID: <WO__9821648A1_I_>

Applications to speed up the synchronization process. Some Applications provide information for keeping track of which records were changed, deleted, or added since the last synchronization. The invention uses these features to speed up the synchronization process by retrieving only those records which have been changed or added since a previous synchronization.

The invention provides a computer implemented method and a computer program for synchronizing a first and a second database. Based on data reflecting whether the records of the first database have been added or changed since a previous synchronization, it is determined whether the records of the first database have been changed or added since the previous synchronization. If one of the records of the first database has not been changed or added since the previous synchronization, a synchronization with records of the second database is performed using a record representative of the one record at the time of a previous synchronization. The representative record is stored in a history file which contains records reflecting the contents of records of the databases at the time of a previous synchronization.

Preferred embodiments of this aspect of the invention may include one or more of the following features. The data provided for keeping track of whether the records of the first database have been added or changed since a previous synchronization may be database generated data, stored in the records of the first database.

The computer generated data indicates the most recent date and time of when a record was created or changed. The computer generated data includes a flag set when a record is created or changed.

The first database provides further database generated data indicating which records were deleted

IOCID: <WO___9821648A1_I_>

database located on a first computer and a second database located on a second computer. At the first computer, it is determined whether a record of the first database has been changed or added since a previous synchronization, using a first history file located on the first computer comprising records representative of records of the first database at the completion of the previous synchronization. If the record of the first database has not been changed or added since the previous synchronization, the first computer sends the second computer information which the second computer uses to identify the record of the first database to be unchanged.

The embodiments of this aspect of the invention may include one or more of the following features.

A second history file may be located on the second computer. The second history file contains records representative of records of the first database at the completion of the previous synchronization, where one of the representative records represents the record of the first database determined to be unchanged. Then, at the second computer, a synchronization of the second and first databases is performed using the one of the representative records.

The information sent from the first computer to the second computer can be used to locate the one of the representative records in the second history file. The second history file can store information in relation to the representative records and the one of the representative records in the second history file can be identified from that stored information. Additionally, the information sent from the first computer to the second computer can include information that matches the information stored in relation to the one of the representative records in the second history files.

XOCID: <WO___9821648A1_I_>

computers, the time needed to synchronize the two databases is decreased

Also, when transmitting data from one computer to another, using a content based code, that requires less bandwidth for being transmitted and nonetheless identifies a record, results in a slow data transfer links being used more efficiently.

The invention may be implemented in hardware or software, or a combination of both. Preferably, the technique is implemented in computer programs executing on programmable computers that each include a processor, a storage medium readable by the processor (including volatile and non-volatile memory and/or storage elements), at least one input device, and at least one output device. Program code is applied to data entered using the input device to perform the functions described above and to generate output information. The output information is applied to one or more output devices.

Each program is preferably implemented in a high level procedural or object oriented programming language to communicate with a computer system. However, the programs can be implemented in assembly or machine language, if desired. In any case, the language may be a compiled or interpreted language.

Each such computer program is preferably stored on a storage medium or device (e.g., ROM or magnetic diskette) that is readable by a general or special purpose programmable computer for configuring and operating the computer when the storage medium or device is read by the computer to perform the procedures described in this document. The system may also be considered to be implemented as a computer-readable storage medium, configured with a computer program, where the storage medium so configured causes a computer to operate in a specific and predefined manner.

)OCID: <WO__9821648A1_I_>

Figure 13 is pseudocode for a host segment of a synchronization program when loading records from and unloading records to the remote database, when the database assigns unique IDs.

Figure 14 is pseudocode for a remote segment of a synchronization program when loading records from and unloading records to the remote database, when the database does not assign unique IDs.

Figure 15 is pseudocode for a host segment of a synchronization program when loading records from and unloading records to the remote database, when the database assigns unique Ids.

Description

Briefly, a synchronization program 100 (Fig. 1) running on a computer loads records of two databases, e.g. a scheduling database on the computer and another one on a remote handheld or notebook computer. The synchronization program then synchronizes the records of those databases using a history file that contains records representative of the records of the two databases at the end of a previous synchronization. The synchronization program uses the history file to determine, for example, which records have been changed, added or deleted since the previous synchronization and which records of the two databases correspond to one another.

If one of the databases has the capability to provide database generated information or data which can be used to determine, for example, whether a record has been changed, added, or deleted since a previous synchronization, the synchronization program uses that information to determine whether a record has been changed, added, or deleted. Of course, that database generated information is less than the whole record of

IOCID: <WO__9821648A1_I_>

previous synchronization. Using the history file to reconstruct the record instead of loading the record can result in significant saving of time - where for example the data transfer link between the two computers is slow - since typically a majority of records in databases are unchanged records. The synchronization program thereby increases the efficiency of performing synchronization between two databases.

In order to better understand embodiments the invention described herein, we will briefly describe the overall structure of a synchronization program and the method it uses to synchronize databases, both of which are described in more detail in the following commonly owned U.S. patent applications, incorporated in their entirety by reference, "Synchronization of Recurring Records in Incompatible Databases", Serial no. 08/752,490, filed on November 13, 1996 (hereinafter, "application '490"); "Synchronization of Databases with Record Sanitizing and Intelligent Comparison," Serial no. 08/749,926, filed November 13, 1996 (hereinafter, "application '926"); "Synchronization of Databases with Date Range," Serial no. 08/748,645, filed November 13, 1996 (hereinafter, "application '645"). We will then describe specifically how the synchronization program operates when at least one of the databases provides information indicating that records have been changed, added or deleted since a previous synchronization.

Fig. 1 shows the relationship between the various modules of an embodiment of synchronization program 100. Translation Engine 1 comprises Control Module 2 and Parameters Table Generator 3. Control Module 2 is responsible for controlling the synchronizing process by instructing various modules to perform specific tasks on the records of the two databases being synchronized. (Fig. 3 shows the steps taken by this module.) The

)OCID: <WO___9821648A1_I_>

designating a priority for a "to do" item should be limited to 1, 2, or 3. Sanitizing a record is to change the content of the fields of a record of one database to conform to the rules of data value of another database. Rules of data value and sanitization are described in detail in the '490, '926 and '645 applications.

In the described embodiment, the modules of A_Translator 5 are designed specifically for interacting with A_database 13 and A_Application 17. Their design is specifically based on the record and field structures and the rules of data value imposed on them by the A_Application, the Application Program Interface (API) requirements and limitations of A_Application 17 and other characteristics of A_Database and A_Application. The same is true of the modules of B_Translator 9. These Translators are not able to interact with any other databases or Applications. They are only aware of the characteristics of the database and the Application for which they have been designed. Therefore, in the preferred embodiment, when the user chooses two Applications for synchronization, the Translation Engine chooses the two Translators which are able to interact with those Applications. In an alternate embodiment, the translators can be designed as table-driven codes, where a general Translator is able to interact with a variety of Applications and databases based on supplied parameters.

Fig. 3 is the pseudocode for the described embodiment of Control Module 2 of the Translation Engine 1. We will use this pseudocode to generally describe the steps taken when synchronizing two databases performed by the synchronization program 100. Control Module 2 first instructs the Parameter Table Generator 3 of Translation Engine 1 to create Parameter_Table 4 (Step 100). In step 102, the Translation Engine instructs synchronizer 15 to

IOCID: <WO__9821648A1_I_>

module 7 of the A_Translator 5, synchronizer 15 maps that record using an A→B_Map before writing the record into the next available spot in the Workspace. Since the A_records are mapped into the B_Record format, when the B_Sanitizer is instructed by Control Module 2 to begin sanitizing those records and starts asking for them from synchronizer 15, they already have the B_Database format. Therefore, synchronizer 15 does not need to map them before sending them to the B_Sanitizer module 12 of the B_Translator 19. For the same reason, there is no need for them to be mapped once they are sent back by the B_Sanitizer after having been sanitized. However, in the case of B_records, they must be mapped using a B→A map prior to being sent to A_sanitizer and then mapped back using an A→B map. At this point, all records are loaded into the Workspace.

Synchronizer 15 then performs a Conflict Analysis and Resolution ("CAAR") procedure on the records in the Workspace, which procedure is described in detail in the '490, '926 and '645 applications. Briefly, during this process, synchronizer 15 compares the records in the workspace and determines what synchronization actions should be taken. Synchronizer 15 processes the records, including comparing them to one another, in order to form them into groups of related records called corresponding item groups (CIGs). Each CIG may comprise at most one recurring or a group of related nonrecurring records from each of the databases and the history file. After forming CIGs from all records of the two databases, synchronizer 15 compares the records in each CIG with one another, determines their differences, and decides what synchronization action should be taken. Synchronization actions with respect to a record include updating, deleting, adding, or not modifying that record. For example, if after comparing the records in a CIG,

DOCID: <WO__9821648A1_I_>

instructs the A_Translator to unload the A_Records from the Workspace (step 111). This unloading is done in the same way as it was done by the B_Translator. Control Module 2 next instructs synchronizer 15 to create a new history file (step 112). During unloading, the Unloader module of the A_Translator uses the A→B map the map the records in the Workspace into the format of A_database records.

At this point Synchronization is complete.

The speed of the above synchronization process can be improved in the case of databases which provide data for keeping track of which records have been changed, added, deleted, or left unchanged since a previous synchronization. As previously described, there are generally two types of such databases: "medium synchronization" and "fast synchronization" databases.

A "fast synchronization" database is a database which provides information regarding changes, deletions, and additions to its records from one synchronization to the next. Synchronization program 100 can use this information to speed up the synchronization process because records which have not been changed are not loaded from the database. Since the majority of records loaded by regular Translators are typically unchanged records, far fewer records are loaded from the database into Workspace than would otherwise be the case.

Certain features are required for a database to be a fast synchronization database. The database assigns each record of the database a unique ID and provides a mechanism for keeping track of which records are added, changed, or deleted from synchronization to synchronization, including a list of deleted records.

There are at least two ways to keep track of additions, changes, and deletions in a database.

DOCID: <WO__9821648A1_I_>

fast synchronization translator provides deleted records, which the regular Translators does not.

In order for such databases to be synchronized without resorting to treating them as regular databases (necessitating loading all records), synchronizer 15 transforms fast synchronization database records from the Translator into equivalent regular database records. These transformed records are then used by synchronizer 15 in the synchronization process. There are two transformations which are necessary. First, synchronizer 15 needs to transform deleted records received from the fast synchronization Translator into a regular database deletions. Second, synchronizer 15 needs to transform lack of output by the fast synchronization Translator into unchanged records.

Synchronization program 100 performs these transformations by using the history file. When the databases are synchronized for the first time, all records in the fast synchronization database are loaded into the history file. As changes, additions, and deletions are made to the fast synchronization database, during each of the subsequent synchronization, the same changes, additions, and deletions are made to the history file. Therefore, the history file at the end of each subsequent synchronization contains a copy of the data in the fast synchronization database.

When a fast synchronization database Translator supplies no input for a unique ID history file record, synchronizer 15 finds (i.e. identifies) the corresponding history file record in Workspace 16, copies it into Workspace 16, and treats the copied record as if it were loaded by the fast synchronization translator itself.

Referring to Fig. 4, steps 1050-1051, synchronizer 15 first verifies that there is an appropriate history file. Because synchronization of fast synchronization

)OCID: <WO__9821648A1_I_>

the '490, '926, and '645 applications, is performed on the records. In general terms, during orientation analysis, deletions and changes to fast synchronization database records are joined with their history file counterparts in unique ID bearing CIGs (step 1107). These are CIGs in which at least one of the records is assigned a unique ID. Synchronizer 15 may treat these CIGs differently than other CIGs to improve the efficiency of the synchronization program, for example, by matching them to history file records based on their unique ID value instead of the content of record, as described in detail in applications '490, '926, and '645.

All history file records and their CIGs are now examined. If there is no corresponding record from the fast synchronization database, it means that the record was unchanged. A clone of the record is made, labelled as being from a fast synchronization database, and joined to the history file record's CIG. At this point the deleted fast synchronization database records marked as deleted are removed from CIGs (step 1109). The fast synchronization records marked as changed are joined in doubleton CIGs (i.e. CIGs entry records from two databases). Those marked as additions are singletons. At this point, the synchronization can proceed as if record of a unique ID bearing regular database were just loaded into Workspace 16.

Whenever records are loaded from a fast synchronization database, all records are loaded so that at the end of synchronization the history file will be the same as the fast synchronization database. Therefore, referring to Fig. 5, in order to perform date range limited synchronization, synchronizer 15 marks the records which fall outside the current and the previous date ranges. A record marked as an added, or during synchronizing from scratch, a record that falls outside

)OCID: <WO__9821648A1_I_>

fast synchronization Translator waits for an acknowledgement from synchronizer 15 that the record has been accepted.

In the case no such acknowledgement is received for a record, the Translator needs to be able to provide that record again to synchronizer 15. If the database allows resetting individual Dirty bits, the Translator merely does not set that bit. If not, the Translator keeps a separate file in which it keeps a record of which fast synchronization records were not accepted. The file may contain the unique IDs of those records. The Translator then uses that file to provide synchronizer 15 with those records during the next synchronization.

We will now describe synchronization in the case of medium synchronization databases. Medium synchronization databases have more limited capabilities than fast synchronization databases for keeping track of addition, deletions, or changes. Medium synchronization database do not keep track of deletions. They however still have the capability to provide information regarding what records were added or modified since a previous synchronization. This information typically takes the form of date and time stamps stored with the record or a Dirty bit, as previously described.

In the case of medium synchronization databases, the Translator provides synchronizer 15 with information indicating which records have been added/changed and which records are unchanged. Based on this information and the history file, synchronizer 15 determines which records were deleted. Synchronizer 15 assumes that those records in the history file for which no matching unique ID was obtained from the database are deleted records and flags them as such. In identifying a record as having been deleted from the database, synchronizer 15 can easily determine what synchronization action should be

IOCID: <WO__9821648A1_I_>

replicates the data of the unchanged record using the content of the history file, in the same manner as in the case of fast synchronization databases. By relevant, we mean data in the fields that are synchronized.

If a record is not unchanged, then the record has either been modified and added since the previous synchronization. The translator sets value of the hidden field as 'C' which indicates to synchronizer 15 that the record was either added or changed (step 1206). The translator then loads the changed or added record, from the database, field by field (steps 1207-1209). The translator then loads the record into the workspace (step 1211).

After the translator has loaded all the records in the database, there are in essence two logical lists in workspace 16. One is the list of records identified as changed/added since a previous synchronization. Second is the list of records identified as unchanged since a previous synchronization. Synchronizer 15 compares the unique IDs of the records in these two lists with the unique IDs in the history file (step 1213). (In the case of date range limited synchronization, the whole database would be loaded but only those records which are in the date range will be synchronized, as was the case for fast synchronization databases.) Based on the comparison, synchronizer 15 determines which records in the history file are no longer present in the database. Synchronizer 15 determines that these records have been deleted and sets the hidden field value for these records as 'D' (step 1214). Synchronizer 15 processes these records in the same manner as when synchronizer 15 receives an indication from a fast synchronization database that a records has been deleted. The synchronizer also compares the unique IDs changed records to the unique IDs of the records of the history file (step 1215). The

OCID: <WO___9821648A1_I_>

database for records that have date and time stamps that are subsequent to the date and time stamp of the last synchronization. Translator determines these records to be changed or added since the last synchronization. The rest of the processing by the Translator and synchronizer are the same as the embodiment described in reference to Fig. 6.

In the case of those databases that provide Dirty bits instead of date and time stamps, the Translator determines which records have their Dirty bits set. Those that have their Dirty bit set are changed or added records while those that do not have their Dirty bits set are unchanged records. The rest of the processing by the Translator and synchronizer are the same as the embodiment described in reference to Fig. 6.

Fig. 8 shows another embodiment for synchronizing a medium synchronization database. In this case, for each record (step 1350), the translator loads the unique ID and the date and time stamp or Dirty bit, as may be the case. If the record has not been modified or added since the previous synchronization (step 1353), the translator calls a special function (PutUnchangedRecord) of synchronizer 15 and also supplies the unique ID of the unchanged record to the synchronizer (step 1354). In response to the function call, synchronizer 15 searches the history file for the supplied unique ID (step 1355) and clones the matching history file record (step 1356).

If the record has been modified or added since the previous synchronization (step 1357), the record is loaded from the database (step 1358-1360) and loaded into the workspace (step 1361). Synchronization then proceeds as if the records of a regular database (i.e. a database that does not provide information that may be used to keep track of whether records of the database have been changed, added or deleted) are loaded, as described

IOCID: <WO__9821648A1_I_>

programs other than a database, as in the case of, for example, general purpose computers such as desktop and notebook computers, or handheld computers having sufficient memory and processing power. In such a case, the synchronization program may be distributed between the two computers so as to, for example, increase the efficiency of using of a slow data transfer link between the two machines.

Briefly, at remote computer 22, remote segment 26 of the synchronization program loads records of remote database 13. Remote segment 26 then determines which records of the remote database have been changed/added, deleted or left unchanged since a previous synchronization. If the remote database assigns unique identification codes (i.e. unique ID) to its records, remote segment 26 can further differentiate between records than have been added and those than have been changed since the previous synchronization. Remote segment 26 uses a remote history file 30 which stores data representing or reflecting the records of the database at the completion of the previous synchronization. This data may be a copy of remote database 13. It may also be hash numbers for each of the records of the remote database. If the remote database assigns unique IDs, the remote history file may contain those unique IDs together with the hash numbers of the records corresponding to the stored unique IDs.

Remote segment 26 sends those records of the remote database that have been changed or added to the host segment or the host computer. However, the remote segment does not send the unchanged or deleted records to the host computer. Instead, the remote segment sends a flag indicating the status of the record (e.g. unchanged or changed) and some data or information that uniquely identifies the record to the host segment. This data or

)OCID: <WO___9821648A1_I_>

components of the synchronization program, such as the name of the databases being synchronized and user preferences. Fig. 11 is the pseudocode of the steps taken by this module. The Synchronizer 15 has primary responsibility for carrying out the core synchronizing functions. It is a table-driven code which is capable of synchronizing various types of databases whose characteristics are provided by control module 2. The Synchronizer creates and uses a host workspace 16 (shown in detail in Fig. 2), which is a temporary data array used during the synchronization process.

A host translator 9 includes two modules: a reader module 10 which reads the data from the host database 14 and an unloader module 10 which analyzes and unloads records from the host workspace into the host database 14. Remote segment 26 also has similar modules for reading and unloading data from the remote database. The remote segment is designed specifically for interacting with remote database 13. The design of the remote segment is specifically based on the record and field structure of the remote database and remote database's Application Program Interface (API) requirements and limitations and other characteristics of the remote database. Similarly host translator 9 is designed specifically for the host database. The remote segment and host translator are not able to interact with any other databases or Applications. They are only aware of the characteristics of the databases for which they have been designed. In an alternate embodiment, the host translator and the remote segment can be designed as a table-driven code, where a general Translator is able to interact with a variety of databases based on the parameters supplied by, for example, the Control Module 2. It should be noted that the remote segment and host

DOCID: <WO__9821648A1_I_>

operation of the control module as if a history file exists and will be used.

Once the History File is loaded into the host workspace, Control Module 2 instructs host translator 13 to load the host database records (step 403). Host Reader module 11 of the host Translator reads the host database records and sends them to the Synchronizer for writing into the host workspace.

Control Module 2 then instructs remote segment to send the records of the remote database (step 404). Remote segment 26 reads the remote database records and sends them to Synchronizer 15 for writing into the host workspace. The actions taken by the synchronizer and the remote segment in response to step 404 will be described in detail in reference to Figs. 12-15, below.

Records in the host workspace are stored according to either the host database or the remote database data structures. Therefore, as synchronizer 15 receives each record, the Synchronizer maps that record using the appropriate record map (i.e. either a remote database to host database record map or a host database to remote database record map) before writing the record into the next available spot in the host workspace. Mapping may be performed by other modules, e.g. the remote segment. The records may also be "translated", i.e. cast into a format which synchronizer can use (a "translation" method is described in the '390 patent). For example, a date stored as "April 1, 97" may be translated into a format preferred by the synchronizer, e.g. "4-1-97".

Control module 2 then instructs the Synchronizer to perform a Conflict Analysis and Resolution ("CAAR") procedure on the records in the host workspace (step 405), which procedure is described in detail in the following applications of the assignee hereof, Puma Technology, Inc. of St. Jose, California, incorporated by

DOCID: <WO___9821648A1_I_>

record with the other database. We will describe below in detail the steps performed by the synchronizer and the remote segment in response to the output of CAAR as the output relates to the remote database.

Once Synchronizer 15 finishes performing CAAR on the records, the records may be unloaded or written into their respective databases, including any additions, updates, or deletions. However, prior to doing so, the user is asked to confirm proceeding with unloading (steps 108-109). Up to this point, neither the databases nor the History File have been modified. The user may obtain through the Control Module's Graphical User Interface (GUI) various information regarding what will transpire upon unloading.

If the user chooses to proceed with synchronization and to unload, the records are then unloaded in order into the host database, the remote database and the History File. The Synchronizer in conjunction with the host translator and the remote segment perform the unloading for the databases. Synchronizer 15 creates a host history File and unloads the records into it. Control Module 2 first instructs the host translator to unload the records from host workspace into the host database. Following unloading of the host records, Control Module 2 instructs the synchronizer and the remote segment to unload the remote records from the host workspace (step 409). We will describe in detail below, in reference to Figs. 12-15, the specific actions taken by Synchronizer 15 and remote segment 26 in order to unload data from the host workspace into the remote database and the update remote history file 28. Control Module 2 next instructs the Synchronizer to create a new History File (step 112). At this point Synchronization is complete.

)OCID: <WO___9821648A1_I_>

implemented with any synchronization program that is able to synchronize such databases.

Generally, the remote segment sends the host segment, over the data transfer link, only the content of those records that have been changed or newly added. As for unchanged records, the history file contains all necessary information to recreate or synchronize those records, if needed. Therefore, it is not necessary to transfer those records to the host segment. Only some data or identification code that uniquely identifies the record to the Synchronizer need be transferred for such a record. Since the majority of records are typically unchanged records, not transferring them over the slow data transfer link improves the efficiency of the synchronization process.

After all necessary information has been transferred to the host segment, the Synchronizer synchronizes the databases. Following synchronization, the host segment transfers information necessary to update the remote database and the remote history file to the remote segment. The remote segment then updates its history file and the remote database.

Since both the host and remote segments rely heavily on history files to enable distributed synchronization, it is important that the host and remote segments use history files that correspond to one another, i.e. both contain records corresponding to a previous synchronization of the same two databases. In the described embodiment, the remote and host history files are named using a common naming convention. The name of a file is made up of six components:

1) Name or ID of the host computer, which may be an assigned name such as an assigned GUID in the case of operating systems by Microsoft Corporation of Redmond,

)OCID: <WO___9821648A1_I_>

the remote segment finds a remote history file that matches the host history file (i.e. a remote history file that matches the host history file) (step 502), then the remote segment examine the date and time stamp of the host and remote history files. If the date and time stamp in the remote history file matches the one in the host history file (step 503), then the remote segment determines that two history files correspond to one another. Hence, the remote segment loads the remote history file into the remote workspace.

In general, if matching history files do not exist on the remote and host computers, the remote segment transfers all remote database records to the host computer. Therefore, if the name of the host and remote history files match but the date and time stamps do not match (step 505), then the remote segment assumes that remote history file is not the correct remote history file to be used. The remote segment removes that history file (step 506) and transfers all remote database records to the host computer (step 507). If no remote history file matches the host history file (step 508), then the remote segment assumes an appropriate remote history file does not exist. The remote segment transfers all the records to the host computer (step 509). To transfer all the records in the above steps, the remote segment first loads and stores all records of the remote database in the remote workspace. The remote segment then transfers all records in the remote database to the host segment. If remote segment transfers all the records of the remote database to the host segment in either step 504 or 509, then the remote will go to step 528. It should be noted that the host segment will use the host history file, if one exists, to perform the synchronization.

If an appropriate remote history file exists - i.e. conditions of steps 501 and 504 are satisfied - the

OCID: <WO__9821648A1_I_>

before comparing the data, with relatively low risk inaccurate comparison. We have also use hash numbers as a unique identification code, which will be described in the second embodiment.

The remote segment uses the remote history file to determine whether a record has been changed, deleted, or added since a previous synchronization. Therefore, for records that are unchanged, which typically constitute the majority of records in a database, the remote segment sends information that the host segment can use to identify the matching records in the host history file. That matching history file record contains the same data as necessary to use for synchronization as that on the remote database since the record is unchanged. Therefore, there is no need to send the whole record. In essence, the remote segment uses the remote history file to filter out information that is already contained in the host history file and sending only those records that have been changed or added. In some embodiments, the remote history file may contain all the field values of the records of the remote database. In those embodiments, the remote segment can determine not only which records have been changed but more specifically which field values have been changed. In that case, the remote segment can determine and then send only those field values that have been changed, further increasing the efficiency of using the slow data transfer link.

We will now describe this process in detail. In the described embodiment, for each record of the remote database (step 515), the remote segment loads the field values, including the unique ID, of the record into the remote workspace (step 512). As the records are loaded, they are translated (e.g. "translated" as described in the '390 patent) into a universal format for the remote workspace. The records will be translated back into the

)OCID: <WO___9821648A1_I_>

marks the record as unacknowledged (step 521), the purpose and function of which we will now briefly describe and is also described in the '490, '926 and '645 applications.

Generally, the remote segment does not change an entry in the remote history file, until it receives an instruction indicating that the host segment has synchronized and updated the host database with that record. This is done so that if for any reason (e.g. user does not want to update that record of the host database as described above) the host database is not synchronized with that record, the remote segment will not treat that record as unchanged during the next synchronization. The acknowledgement may take the form of an "acknowledgment" flag or an "action" instruction which instructs the remote segment to add, update, or delete that record of the remote database, as described above. Therefore, for each changed and deleted record, the remote segment creates a new entry and marks the entry as "unacknowledged". If an "acknowledgment" flag is received, the old history file record is deleted. If an "acknowledgement" flag is not received, the new workspace entry is deleted. The steps will be described further below.

If in step 515 the remote segment determines that the unique ID of the loaded record does not match any of the unique IDs stored in the records of the history file (step 521), the remote segment assumes that the record loaded from the remote database has been newly added. Therefore, the remote segment sends the host segment a copy of the field values of those fields of the record to be synchronized (which may be all or less than all the fields) together with an "added" flag (step 524). As in the case of a changed record, the remote segment creates a new remote workspace entry and enters the unique ID and

DOCID: <WO__9821648A1_I_>

In order to synchronize the remote database with the host database, the Synchronizer transforms information from the remote segment into regarding unchanged records into equivalent regular database records. These transformed records are then used by the Synchronizer in the synchronization. Essentially, the synchronizer transforms and uses the information sent by the remote segment to identify a record in the history file that is a copy of the field values of the unchanged remote database record. In the described embodiment, the synchronizer also copies that history file record and flags as being the remote database record.

The described embodiment uses the host history file to perform this transformation. At the beginning of a first synchronization between the two databases, all records in the remote database are loaded into the host history file. As changes, additions, and deletions are made to the remote database, during each subsequent synchronization, the same changes, additions, and deletions are made to the host history file. Therefore, the host history file at the end of each synchronization will contain a copy of the relevant content of the remote database after synchronization. By relevant, we mean data in the fields that are synchronized. For example, it may be the case that the host history file contain data in fields that are not synchronized. Moreover, if the records of the remote are mapped or recast into another format (e.g. "translated" as described in the '390 patent) the records of the history file contain a copy of the records of the database, as mapped, translated, or both. The Synchronizer uses the mapped or translated records for synchronization. Therefore, it only needs the mapped or translated copy of the unchanged record. In other embodiments, the host history file may contains copies of all the records exactly as they are in

XOCID: <WO___9821648A1_I_>

As previously described, after the synchronizer has performed CAAR, the user must confirm to proceed with updating the remote database (step 406 in Fig. 11). If the user decides to terminate the synchronization, changes are not made to the host history file or the databases. In the case of the remote database, as described in reference to Fig. 12, the remote segment is waiting for the synchronizer to finish synchronizing. If the user aborts synchronization (step 533), the remote segment discards the remote workspace (step 534), saves the original history file without any changes (step 535), and terminates the process at the remote computer.

If the user confirms to proceed with updating the database (step 406 in Fig. 11) , control module 2 instructs the synchronizer and the remote segment to proceed with unloading the records from the workspace into the remote database. As stated, at this point, the remote segment is waiting for the synchronizer to finish synchronizing (step 532 in Fig. 12). During the synchronization, the synchronizer has determined what "actions" with respect to which record in which database should be taken (update, delete, or add) to complete synchronization. If changes or additions are made to the host database in the case of particular record but no action need be taken with respect to that record in the remote database, the synchronizer determines that an "acknowledgement" should be sent to the remote segment. The synchronizer sends all the actions concerning the remote database together with the associated record to the remote (step 616). The synchronizer then sends the unique ID of those records that require "acknowledgements" to be sent to the remote together with an appropriate flag (step 617).

Referring again to Fig. 12, for each action item or acknowledgement received at the remote segment (step

IOCID: <WO___9821648A1_I_>

history file to end the synchronization of the two databases.

In the first embodiment, which we described above, the remote database assigns unique IDs to its records. We will now describe a second embodiment for the case where the remote database does not assign unique IDs to its records. In such a case, the remote segment provides some information less than all the fields of the records to uniquely identify an unchanged record to the host segment. This information may be a hash value. The host segment uses this information to find and then use the host history file copy of the unchanged remote database record to synchronize the two databases.

To identify a record from the previous synchronization or an unchanged record, the remote segment can use a content based code, that is a code whose value depends on the content of all or a selected number of the fields of a record. In the second embodiment, the remote segment uses hash numbers. Since in the case of an unchanged record, its content has remained the same, its hash number remains the same. The hash number acts as a unique identifier and therefore enables the remote and host segments to identify the unchanged record by its hash code. The hash code can be used to identify a record that is stored in the host history file, since its content remains the same from the end of one synchronization to the time it is updated. It may also be transmitted to identify an unchanged record or an unchanged version of a changed record. A host history file record can in effect be identified using the hash code of that record.

We will describe the operation of this embodiment in reference to Figs. 14 and 15. Steps 701 -711 are the same as steps 501-511 in Fig. 12, described above in reference to the first embodiment. These steps are

IOCID: <WO__9821648A1_I_>

Therefore, the remote segment sends the host segment a copy of the field values of the record, the remote workspace index number, and an "added" flag (step 720). The remote workspace index number makes it easier to perform future search of the remote workspace when data with respect to this record is received. As in the case of changed and added record in the first embodiment, the remote segment also creates a new remote workspace entry and enters hash number value of the record (step 718). The new entry is marked as "unacknowledged" (step 719). It should be noted that although the remote segment treats the record as a new record, the remote segment can not distinguish between an added and a changed record. Therefore, the synchronizer during synchronization does not treat it as a new record. Instead, the synchronizer compares the record to determine whether it matches with any of host history file record which would mean it is a changed record.

After reading all the remote database records and processing them (step 722), the remote segment removes from the remote workspace all entries that have hash numbers that are unmatched (step 723). These entries represent records that have either been changed or deleted since the previous synchronization.

After the remote segment has finished providing data to the host segment, the host segment synchronizes the two databases based on the input from the remote segment. The remote segment waits until the host segment finishes synchronizing and instructs the remote segment in step 409 in Fig. 11 to begin unloading into the remote database (step 724).

Referring to Fig. 15, as in the case of the first embodiment, the synchronizer on the host computer uses the information to identify those records in the host history file that correspond to the unchanged remote

XOCID: <WO___9821648A1_I_>

The synchronizer sends all the actions concerning the remote database together with the associated record and remote workspace index to the remote (step 809). The synchronizer then sends the remote workspace index of those records that require acknowledgements to be sent to the remote together with an appropriate flag (step 810). Therefore, the remote workspace index is used to identify which records in the remote workspace should be "acknowledged".

Referring back to Fig. 14, steps 725-729 are the same as steps 533-537, which were described in reference to the first embodiment. For each action item or acknowledgement received at the remote segment (step 730), the following steps are performed. If the data received indicates an "acknowledgement" or "action" with respect to a record that was sent to the host segment flagged as "added" (step 731), the remote segment marks the new workspace entry that was created in either step 718 as acknowledged (step 732). It should be noted that the remote workspace index number is used to locate the remote workspace entry. Therefore, as previously described, this entry will be written into the history file at the end of the process at the remote segment.

If the received data indicates an action item that tells the remote segment to update, change, or add a remote database record (step 733), the remote segment performs that action with respect to the remote database. The remote segment also updates the remote workspace and marks the entry as "acknowledge" (step 735).

After all the records have been received, the remote segment discards all unacknowledged entries from the workspace, which were newly created entries which were not acknowledged. Therefore, in case of those added or changed records with the user decided not to update the host database with, the remote history file remains

OCID: <WO___9821648A1_I_>

What is claimed is:

1. A computer implemented method of synchronizing a first and a second database comprising:

determining whether the records of the first database have been changed or added since the previous synchronization, based on data reflecting whether the records of the first database have been added or changed since a previous synchronization;

if one of the records of the first database has not been changed or added since the previous synchronization, performing a synchronization with records of the second database using a record representative of the one record at the time of a previous synchronization, the representative record being stored in a history file containing records reflecting the contents of records of the databases at the time of a previous synchronization.

2. The computer implemented method of claim 1, wherein the data comprises database generated data for keeping track of whether the records of the first database have been added or changed since a previous synchronization.

3. The computer implemented method of claim 2 wherein the computer generated data indicates the most recent date and time of when the records were created or changed.

4. The computer implemented method of claim 2 wherein the computer generated data comprises a flag set when the records are created or changed.

)OCID: <WO__9821648A1_I_>

8. The computer implemented method of claim 2 further comprising:

modifying, adding, or deleting the records of the first database based on a result of the synchronization.

9. The computer implemented method of claim 2 further comprising modifying, adding, or deleting records in the history file using results of the synchronization such that the history file contains records reflecting the contents of records of the databases after the synchronization.

10. A computer program, resident on a computer readable medium, for synchronizing a first and a second database, comprising instructions for:

determining whether the records of the first database have been changed or added since the previous synchronization, based on data reflecting whether the records of the first database have been added or changed since a previous synchronization;

if one of the records of the first database has not been changed or added since the previous synchronization, performing a synchronization with records of the second database using a record representative of the one record at the time of a previous synchronization, the representative record being stored in a history file containing records reflecting the contents of records of the databases at the time of a previous synchronization.

11. The computer program of claim 10, wherein the data comprises database generated data for keeping track of whether the records of the first database have been added or changed since a previous synchronization.

)OCID: <WO___9821648A1_I_>

16. The computer program of claim 11 wherein the first database assigns a unique identification data to the records of the first database, the computer program further comprising instructions for:

performing a comparison of the records of the history file and the unique identification data of records in the first database that have been changed or added since the previous synchronization; and

completing synchronization using a result of the comparison.

17. The computer program of claim 11 further comprising instructions for:

modifying, adding, or deleting records of the first database based on a result of the synchronization.

18. The computer program of claim 11 further comprising instructions for modifying, adding, or deleting records in the history file using results of the synchronization such that the history file contains records reflecting the contents of records of the databases after the synchronization.

XOCID: <WO__9821648A1_I_>

20. A computer program, resident on a computer readable medium for synchronizing a first database located on a first computer and a second database located on a second computer, comprising instructions for:

determining whether a record of the first database has been changed or added since a previous synchronization;

if the record of the first database has not been changed or added since the previous synchronization, sending from the first computer to the second computer information which the second computer uses to identify the record of the first database to be unchanged, wherein a history file located on the second computer contains records reflecting the content of records of the first database at the completion of a previous synchronization and said information identifies a record in the history file reflecting the content of the record of the first database; and

synchronizing the first database and the second database using the sent information.

IOCID: <WO___9821648A1_I_>

24. The computer implemented method of claim 23 wherein the second history file stores information in relation to the representative records and wherein the one of the representative records in the second history file can be identified from the stored information.

25. The computer implemented method of claim 24 wherein the information sent from the first computer to the second computer comprises information that matches the information stored in relation to the one of the representative records in the second history files.

26. The computer implemented method of claim 21 wherein the information comprises information identifying records other than the unchanged record.

27. The computer implemented method of claim 21 wherein the information comprises information identifying the unchanged record.

28. The computer implemented method of claim 21 wherein the information comprises information identifying the deleted records.

29. The computer implemented method of claim 21 wherein the information comprise information identifying the added records.

30. The computer implemented method of claim 21 wherein the information comprises a code, the code being based on at least a portion of the content of the record of the first database.

IOCID: <WO__9821648A1_I_>

36. A computer implemented method of identifying a record of a database stored on a first computer to a second computer comprising:

reading a record of the database;

assigning a code to the record of the database, the code being based on at least a portion of the content of the record of the first database;

transmitting the code to the second computer to identify the record to the second computer.

37. The computer implemented method of claim 36 wherein the code comprises a hash number computed based on at least a portion of the content of the record of the first database.

38. A computer program, resident on a computer readable medium for synchronizing a first database located on a first computer and a second database located on a second computer, comprising instructions for:

determining, at the first computer, whether a record of the first database has been changed or added since a previous synchronization, using a first history file located on the first computer comprising records representative of records of the first database at the completion of the previous synchronization;

if the record of the first database has not been changed or added since the previous synchronization, sending from the first computer to the second computer information which the second computer uses to identify the record of the first database to be unchanged.

)OCID: <WO___9821648A1_I_>

45. The computer program of claim 38 wherein the information comprises information identifying the deleted records.

46. The computer program of claim 38 wherein the information comprise information identifying the added records.

47. The computer program of claim 38 wherein the information comprises a code, the code being based on at least a portion of the content of the record of the first database.

48. The computer program of claim 47 wherein the code comprises a hash number computed based on at least a portion of the content of the record of the first database.

49. The computer program of claim 47 wherein the information further comprises a first plurality of records of the first database identified as "changed or added", the program further comprising instructions for using said information to indentify a plurality of the first database as "deleted or changed" since a previous synchronization.

50. The computer program of claim 38 wherein the information comprises a code uniquely identifying the record of the first database.

51. The computer program of claim 50 wherein the unique identification code is assigned by the first database to the record of the first database.

)OCID: <WO__9821648A1_I_>

55. A computer implemented method of synchronizing at least a first and a second database, wherein the first database provides information regarding records which have been added, deleted or modified since a previous synchronization, the method comprising:

storing in a history file records representative of the records of the first database after the previous synchronization;

obtaining information regarding the records which have been modified, deleted, or added since the previous synchronization;

performing a first comparison of the records of the second database with records in history file corresponding to records of the first database that have not been modified, added, or deleted;

completing the current synchronization based on the outcome of the first comparison and the information; and

modifying, adding, or deleting records in the history file using the information obtained from the first database such that the history file contains records representative of the records of the first database after the current synchronization.

DOCID: <WO__9821648A1_I_>

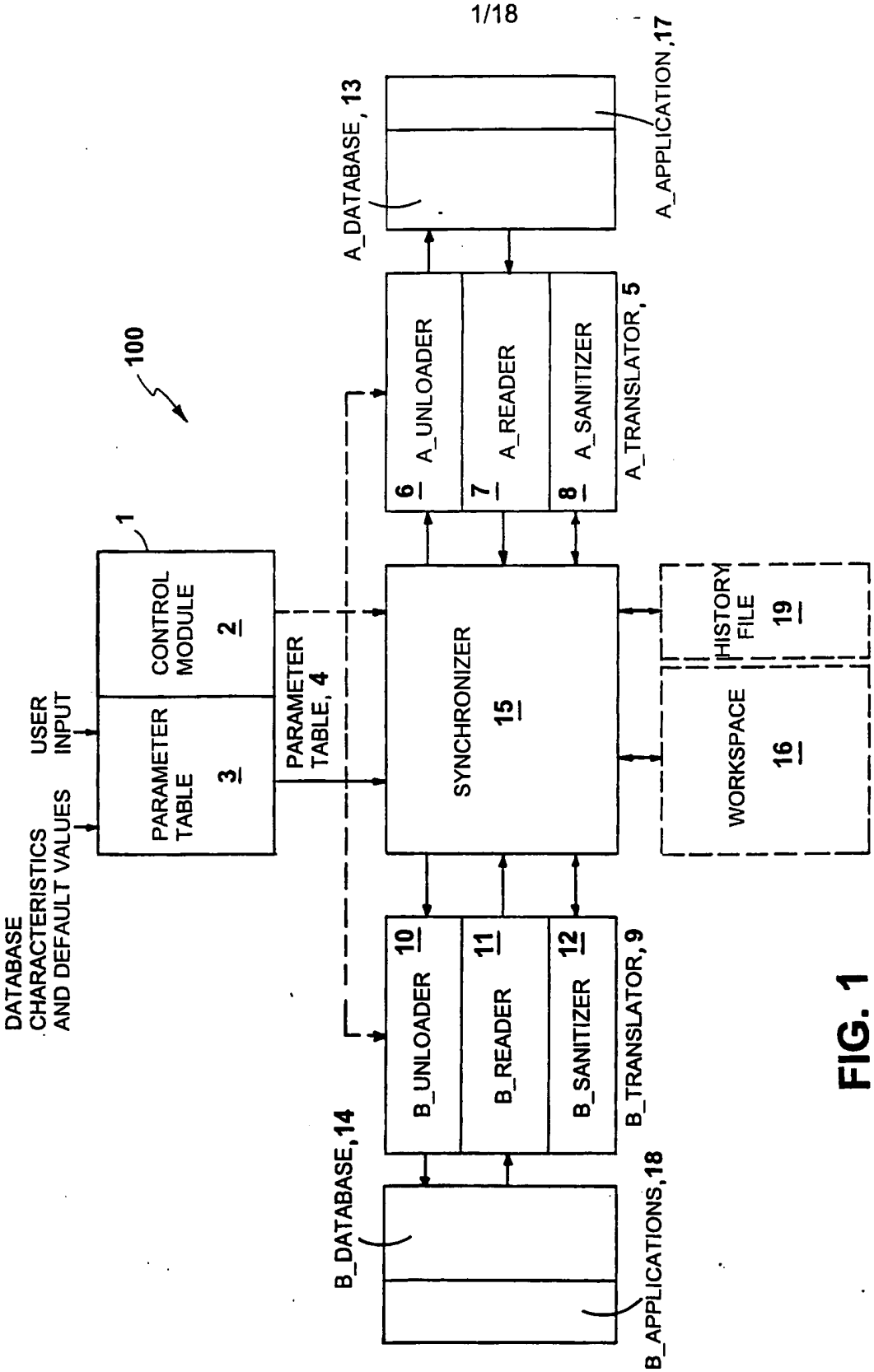


FIG. 1

IOCID: <WO___9821648A1_I_>

IDCID: <WO___9821648A1_I_>

Pseudo Code for Translation Engine Control Module

```
100.  CREATE Parameter_Table from User Input A & B database characteristics and default values
101.  INSTRUCT Synchronizer to initialize itself
102.  INSTRUCT Synchronizer to LOAD the History_File into its WORKSPACE
103.  INSTRUCT B_Translator to LOAD all of B_records from B_Database and SEND to Synchronizer
      (Synchronizer STORES these records in WORKSPACE)
104.  INSTRUCT A_Translator to SANITIZE B_records that were just LOADED (A_Translator USES
      Synchronizer services to read and write records in the WORKSPACE; Synchronizer maps these records
      using the B-A_Map before sending them to A_Translator and maps them back using A-B_Map before
      rewritting them into the WORKSPACE)
105.  INSTRUCT A_Translator to LOAD all of A_records form A_Database and SEND to  Synchronizer
      (Synchronizes STORES these records in WORKSPACE by first mapping them using the A-B_Map and
      them storing in their new form)
106.  INSTRUCT B_Translator to SANITIZE A_records that were just LOADED (B_Translator uses
      Synchronizer services to read and write records in the WORKSPACE)
107.  INSTRUCT Synchronizer to do CAAR (Conflict Analysis And Resolution) on all the records in
      WORKSPACE.
108.  INFORM user exactly what steps Synchronizer proposes to take (i.e. Adding, Changing, and Deleting
      records).  WAIT for User
109.        IF user inputs NO, THEN ABORT
110.  INSTRUCT B_Translator to UNLOAD all applicable records to B_Database.
111.  INSTRUCT A_Translator to UNLOAD all applicable records to the A_Database.
112.  INSTRUCT Synchronizer to CREATE a new History File.
```

FIG. 3

SUBSTITUTE SHEET (RULE 26)

SUBSTITUTE SHEET (RULE 26)

OCID: <WO___9821648A1_I_>

```
1150.  Verify History File
1151.      If verified, Then Proceed as Fast Synch
1152.      If not, Then Proceed as Synchronization from Scratch

1153.  IF synchronization from scratch
1154.      IF record outside of current_date_range THEN MARK record as out-of-range

1155.  If Fast Synch
1156.      Load History File into Workspace
1157.      MARK History File records outside of previous_date_range as Bystander
1158.      Load All Fast Synchronization Records into the Workspace; mapped if necessary.
1159.      SANITIZE Records which are not DELETEs
1160.      Orientation analysis (Fig. 11).
1161.      If Added Fast Synchrnization record is out of current date range THEN MARK Out-Of_Range
1162.      If Changed or deleted Fast Synchronization record in a CIG with Bystander H_Record, MARK
          the Bystander record as Garbage
1163.      For each H_Record, analyze the CIG that the H_Record belongs to.
1164.          If the H_Record's CIG contains no Record from the Fast Synchronization database,
              then make a clone of the H-Item, label it a Fast Synchronization Record, and adding it
              to the H_Record's CIG.
1165.          If H_Record is not a Bystander, THEN Make a clone of H_Record, mark as Fast
```

FIG. 5A

XCOID: <WO__9821648A1_I_>

SUBSTITUTE SHEET (RULE 26)

```
1200  FOR each Record in Database
1201        Get Unique ID
1202        Get Last modified Date/Time
1203              IF Last modified Date/Time Stamp (Less Than or Equal To) Last Sync Date/Time Stamp
1204                    Set Flag to indicate "Unchanged" Record
1205              ELSE
1206                    Set Flag to indicate "Changed/New" Record
1207                    FOR each Field in Record
1208                          Load Field Data into the Workspace
1209                    NEXT
1210              ENDIF
1211        Return record to the Sync Engine [Please explain.what this line does]
1212        NEXT
```

In the synchronizer:

1213 Compare the loaded unique IDs with the Unique ID of all records in the history file
1214 Determine as deleted those records whose unique ID of records in the history file but not in loaded records
1215 Compare unique IDs of records marked as changed to unique IDs of the records of the history file.
1216 Determine as Added those records having unique IDs not present in the history file
1217 Proceed as in fast synchronization to synchronize
1218 Update the history file as for fast synchronization databases
1219 Unload records as for fast synchronization databases

FIG. 6

```
1350.  For each Record in Database
1351.      Get unique ID
1352.      Get Last modified Date/Time or Dirty bit, as applicable
1353.      If the record has not been changed since the previous synchronization then
1354.          Call Synchronizer's PutUnchangedRecord function with unique ID
1355.          In the Synchronizer: in reponse to the PutUnchangedRecord fucntion call, use unique
               ID to find matching History file record using unique ID
1356.          Clone History record
1357.      Else
1358.          For each Field in Record
1359.              Load Field Data
1360.          Next Field
1361.      Put loaded record in the workspace.
1362.  Next Record.

       In the Synchronizer:
1363.  Synchronize the two database as if the records were loaded from a regular (i.e. non-fast synchronization
       database)
```

FIG. 8

DOCID: <WO___9821648A1_I_>

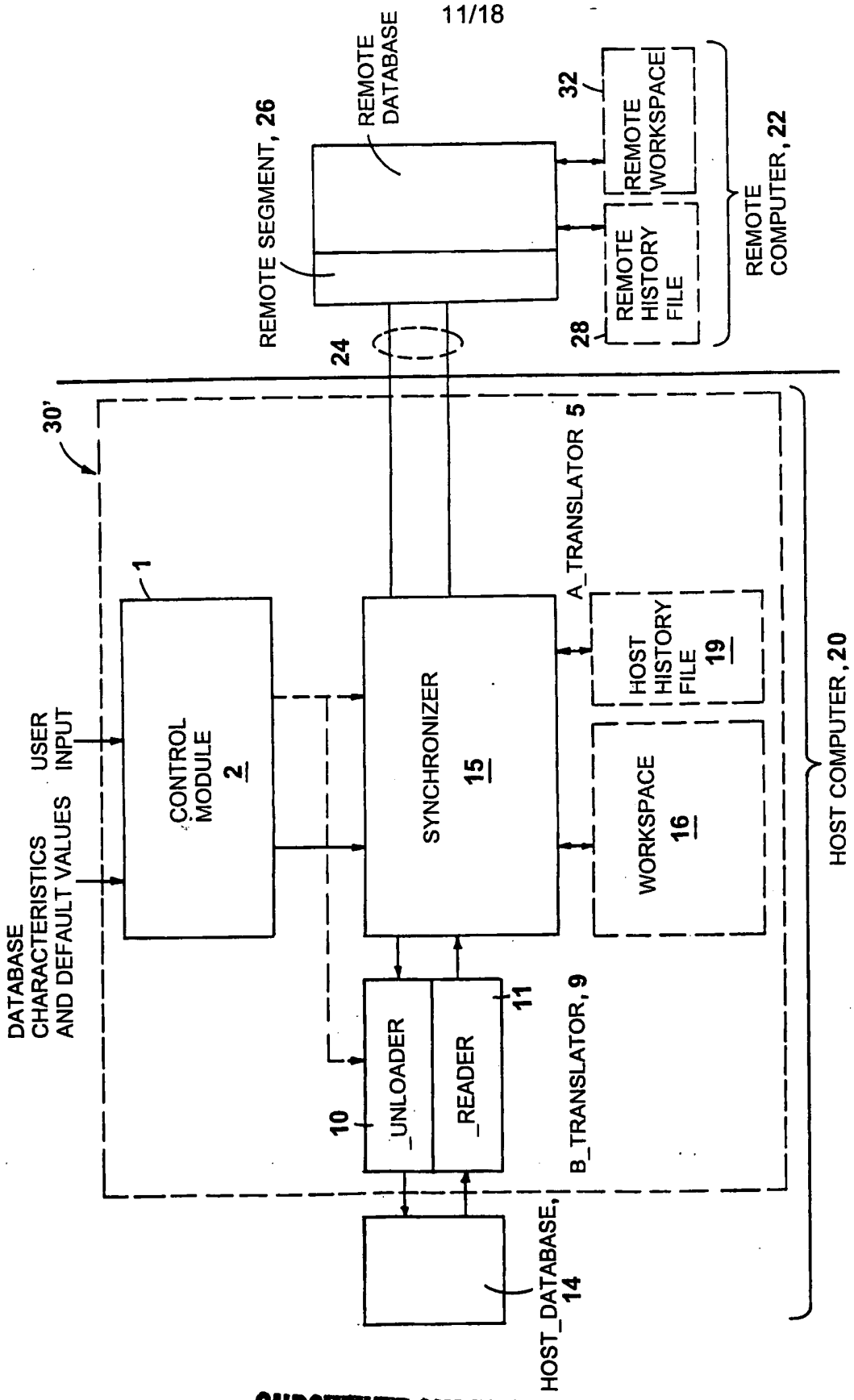


FIG. 10

IOCID: <WO___9821648A1_I_>

DOCID: <WO__9821648A1_I_>


```
501.  INITIALIZE an empty remote workspace
502.  IF there is a remote history file matching the host history file name
503.        IF the remote history file time stamp matches the history file time stamp
504.              LOAD the remote history file into the remote workspace
505.        ELSE
506.              REMOVE the non-matching history file
507.              Proceed with the empty workspace, all records passed to host
508.        ENDIF
509.  ELSE
510.        Proceed with the empty workspace, all records passed to host
511.  ENDIF
512.  FOR each record in the remote database
513.        Translate and load data field values and unique ID into remote workspace
514.        Compute a hash value to represent all translated data values
515.        IF the unique ID matches the unique ID of an existing remote history file entry,
516.              IF the hash value is the same
517.                    Skip this entry, the host will recreate this record from history
518.              ELSE
519.                    Send Unique ID, field values and "Changed" record flag to the host
520.                    Create new workspace entry with same unique ID and new hash value
521.                    This new entry is marked as "unacknowledged"
522.              ENDIF
```

FIG. 12A

SUBSTITUTE SHEET (RULE 26)

XOCID: <WO__9821648A1_I_>

Recreation of "filtered" data by the synchronizer using history data when unique ID's are present.

```
601.    FOR all Added, Changed, and Deleted records transmitted from the remote ,
602.            IF record was flagged "Added"
603.                    Add the new record to the workspace, not correlated with any history at this time
604.            ELSE IF record has been changed
605.                    Find corresponding history item in the workspace by the changed record's Unique ID
606.                    Associate (link) the changed record with this workspace item
607.            ELSE IF record has been deleted
608.                    Find corresponding history item in the workspace by the deleted record's Unique ID
609.                    Associate (link) the delete action with this workspace item
610.            ENDIF
611.    NEXT
612.    FOR all history records not yet matched by a remote Unique ID,
613.            CLONE the missing record data and Unique ID from data in the history file
614.    NEXT
```

Updating the remote computer's remote history file while unloading changes from the workspace to the remote database (when unique ID's are used):

```
615.    FOR all actions or acknowledgments recorded in the workspace for the remote database
616.            SEND all actions with associated record data to the remote
617.            SEND all acknowledgments to the remote with associated unique ID's
618.            UPDATE workspace with unique ID's sent from the remote as the result "Add" actions.
619.    NEXT
```

FIG. 13

SUBSTITUTE SHEET (RULE 26)

```
        host
721.            ENDIF
722.    NEXT
723.    REMOVE any "prior" workspace entries not matched by hash value above
724.    WAIT for host to synchronize the data and for user to confirm results
725.    IF user has aborted the synchronization
726.            The remote workspace is discarded.
727.            The original remote history file remains unmodified.
728.            The process is terminated.
729.    ENDIF
730.    FOR each record "action" or "acknowledgment" received from the host,
731.            IF this is an acknowledgment of a record sent to the host (above) as "added",
732.                    Mark any corresponding, newly created workspace item as "acknowledged"
733.            ELSE IF this is a new action to Add, Update, or Delete a remote database record
734.                    UPDATE remote workspace to reflect the appropriate change
735.                    Mark the updated record as "acknowledged"
736.            ENDIF
737.    NEXT
738.    REMOVE any newly create, but "unacknowledged" entries from the workspace
    UPDATE the remote history file from the remote workspace
```

FIG. 14B

SUBSTITUTE SHEET (RULE 26)

DOCID: <WO__9821648A1_I_>

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US97/20627

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) : G06F 9/00
US CL : 701/1, 100, 200

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 701/1, 100, 200

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

APS
search terms: history file, synchronization, dtabase, date, time, comparison, identifier

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,511,188 (PASUCCI et al.) 23 April 1996, abstract	1-56
Y	US 4,827,423 (BEASLEY et al.) 02 May, 1989, abstract	1-56
Y	US 4,980,844 (DEMJANENKO et al.) 25 December 1990, abstract	1-56
Y	US 5,251,151 (DEMJANENKO et al.) 05 October 1993, abstract	1-56
Y	US 5,444,851 (WOEST) 22 August 1995, abstract	1-56
Y	US 5,463,735 (PASUCCI et al.) 31 October 1995, abstract	1-56

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search	Date of mailing of the international search report
26 JANUARY 1998	25 FEB 1998
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer DAVID JUNG Telephone No. (703) 308-5262

Form PCT/ISA/210 (second sheet)(July 1992)★

XOCID: <WO___9821648A1_I_>

CORRECTED VERSION*

PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 9/00 — A1

(11) International Publication Number: WO 98/21648

(43) International Publication Date: 22 May 1998 (22.05.98)

(21) International Application Number: PCT/US97/20627

(22) International Filing Date: 13 November 1997 (13.11.97)

(30) Priority Data:

08/749,926	13 November 1996 (13.11.96)	US
08/927,922	11 September 1997 (11.09.97)	US
08/964,751	5 November 1997 (05.11.97)	US

(71) Applicant *(for all designated States except US)*: PUMA TECHNOLOGY, INC. [US/US]; 2550 North First Street #500, San Jose, CA 95131 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: BOOTHBY, David, J. [US/US]; 12 Thoreau Drive, Nashua, NH 03062 (US). DALEY, Robert, C. [US/US]; 13 Middle Dunstable Road, Nashua, NH 03062 (US). MARIEN, John, R. [US/US]; 23 Royal Crest Drive, No. 12, Nashua, NH 03062 (US).

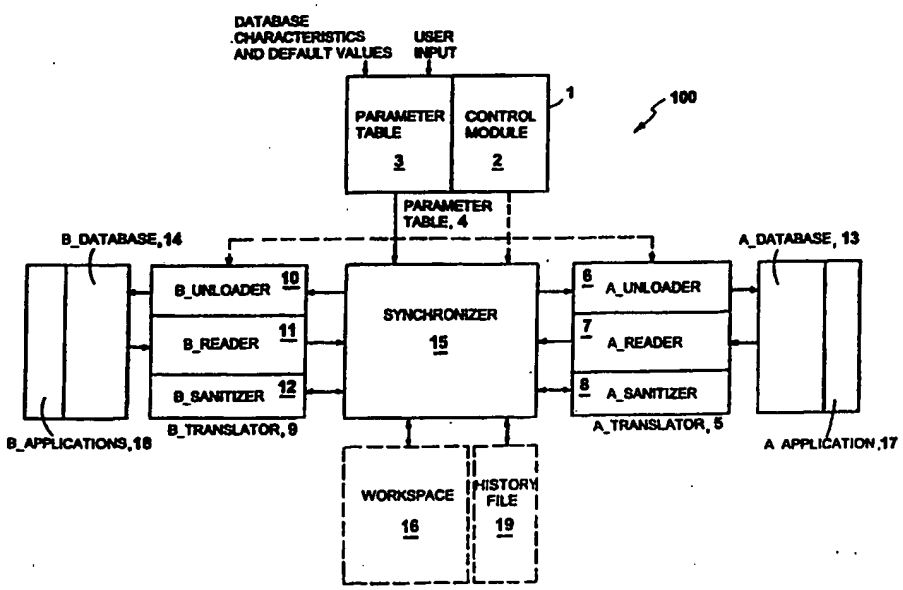
(74) Agent: LEE, G., Roger; Fish & Richardson P.C., 225 Franklin Street, Boston, MA 02110-2804 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZW, ARIPO patent (GH, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published

With international search report.

(54) Title: SYNCHRONIZATION OF DATABASES



(57) Abstract

A computer implemented method and a computer program for synchronizing a first (13) and a second database (14), where data is provided for keeping track of whether the records of the first database have been added or changed since a previous synchronization. This data, for example, may be generated by the first database or by a segment of the synchronization running on a computer on which the first database is located while another segment of the synchroniztion program runs on another computer. Based on the data reflecting whether the records of the first database have been added or changed since a previous synchronization, it is determined whether the records of the first database have been changed or added since the previous synchronization. If one of the first database has not been changed or added since the previous synchronization, a synchronization with records of the second database is performed using a record representative of the one record at the time of a previous synchronization. The representative record is stored in a history file (19) which contains records reflecting the contents of records of the databases at the time of a previous synchronization.

*(Referred to in PCT Gazette No. 34/1998, Section II)

OCID: <WO___9821648A1_IA>

CORRECTED VERSION* | CORRECTED VERSION**

PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 9/00

A1

(11) International Publication Number: WO 98/21648

(43) International Publication Date: 22 May 1998 (22.05.98)

(21) International Application Number: PCT/US97/20627

(22) International Filing Date: 13 November 1997 (13.11.97)

(30) Priority Data:

08/749,926	13 November 1996 (13.11.96)	US
08/927,922	11 September 1997 (11.09.97)	US
08/964,751	5 November 1997 (05.11.97)	US

(63) Related by Continuation (CON) or Continuation-in-Part (CIP) to Earlier Application

US 08/749,926 (CIP)
Filed on 13 November 1996 (13.11.96)

(71) Applicant *(for all designated States except US)*: PUMA TECHNOLOGY, INC. [US/US]; 2550 North First Street #500, San Jose, CA 95131 (US).

(72) Inventors; and

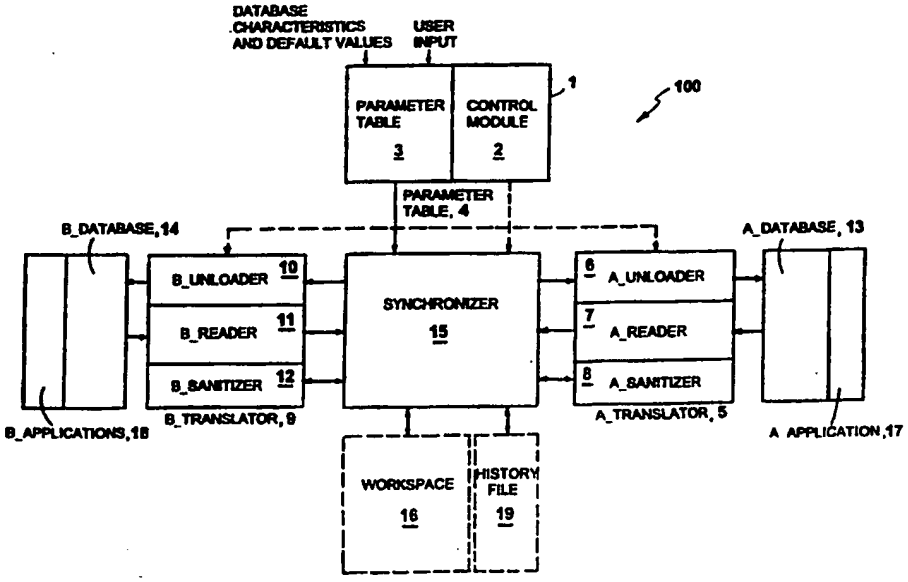
(75) Inventors/Applicants *(for US only)*: BOOTHBY, David, J. [US/US]; 12 Thoreau Drive, Nashua, NH 03062 (US). DALEY, Robert, C. [US/US]; 13 Middle Dunstable Road, Nashua, NH 03062 (US). MARIEN, John, R. [US/US]; 23 Royal Crest Drive, No. 12, Nashua, NH 03062 (US).

(74) Agent: LEE, G., Roger; Fish & Richardson P.C., 225 Franklin Street, Boston, MA 02110-2804 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZW, ARIPO patent (GH, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(54) Title: SYNCHRONIZATION OF DATABASES



(57) Abstract

A computer implemented method and a computer program for synchronizing a first (13) and a second database (14), where data is provided for keeping track of whether the records of the first database have been added or changed since a previous synchronization. This data, for example, may be generated by the first database or by a segment of the synchronization running on a computer on which the first database is located while another segment of the synchroniztion program runs on another computer. Based on the data reflecting whether the records of the first database have been added or changed since a previous synchronization, it is determined whether the records of the first database have been changed or added since the previous synchronization. If one of the first database has not been changed or added since the previous synchronization, a synchronization with records of the second database is performed using a record representative of the one record at the time of a previous synchronization. The representative record is stored in a history file (19) which contains records reflecting the contents of records of the databases at the time of a previous synchronization.